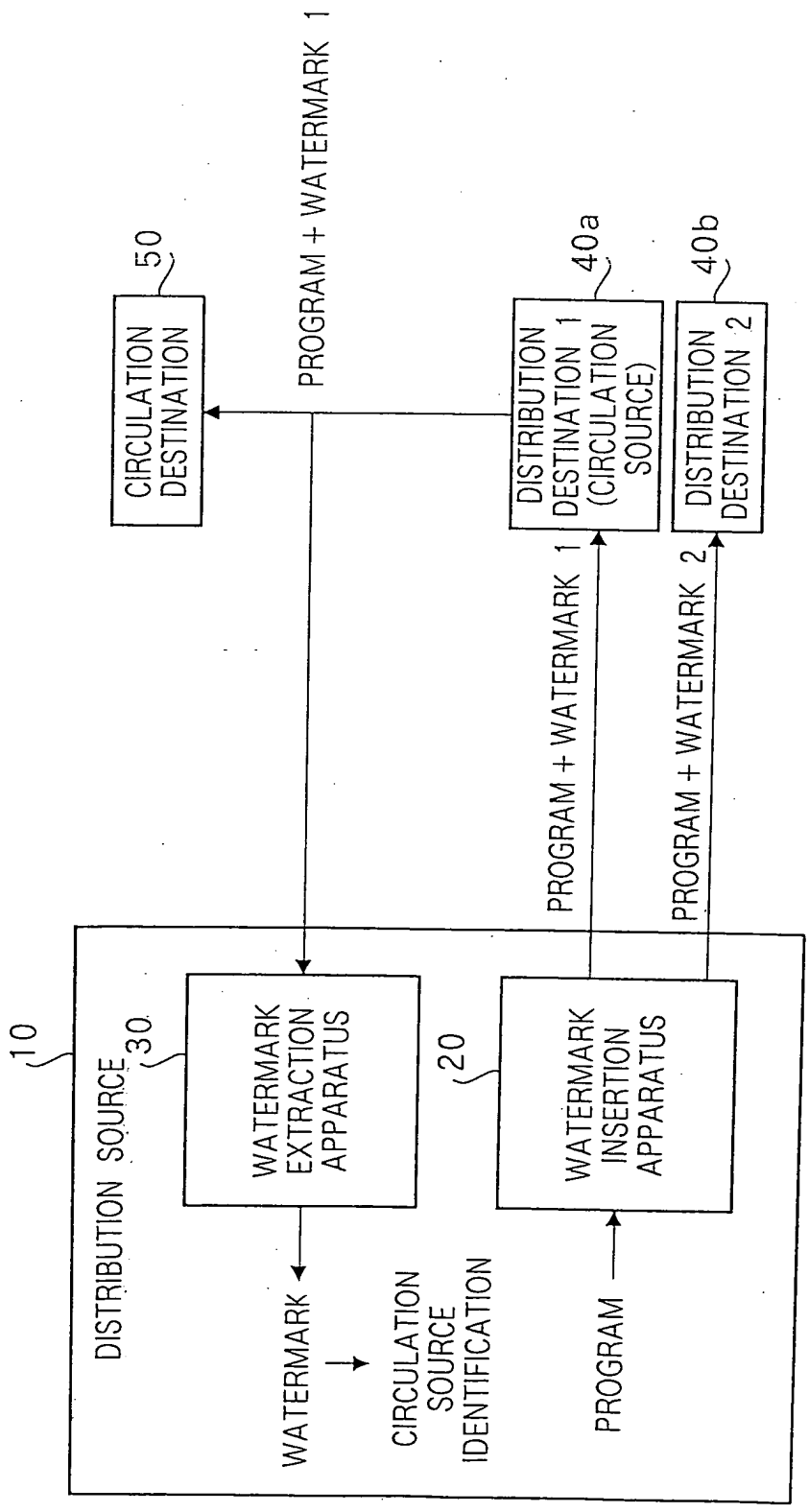
10
DISTRIBUTION SOURCE
30
WATERMARK EXTRACTION APPARATUS
20
WATERMARK INSERTION APPARATUS
WATERMARK
CIRCULATION SOURCE IDENTIFICATION
PROGRAM
PROGRAM + WATERMARK 1
PROGRAM + WATERMARK 2
50
CIRCULATION DESTINATION
PROGRAM + WATERMARK 1
40a
DISTRIBUTION DESTINATION 1 (CIRCULATION SOURCE)
40b
DISTRIBUTION DESTINATION 2

FIG. 1

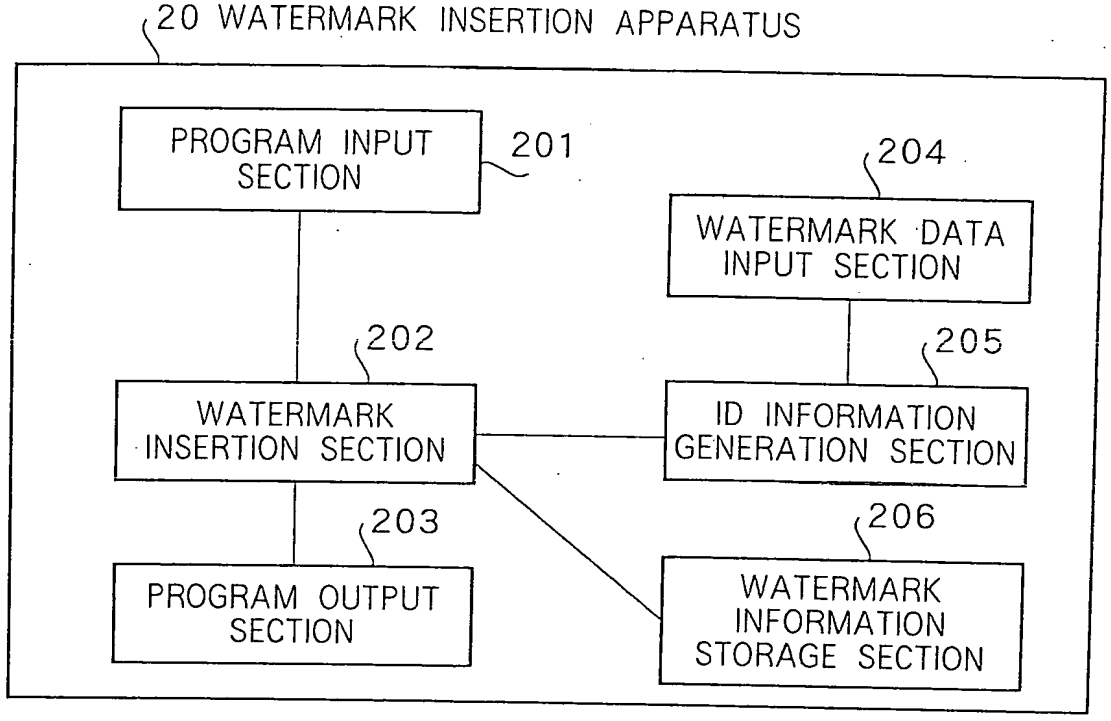
20 WATERMARK INSERTION APPARATUS
PROGRAM INPUT SECTION
201
204
WATERMARK DATA INPUT SECTION
202
205
WATERMARK INSERTION SECTION
ID INFORMATION GENERATION SECTION
203
206
PROGRAM OUTPUT SECTION
WATERMARK INFORMATION STORAGE SECTION

FIG. 2

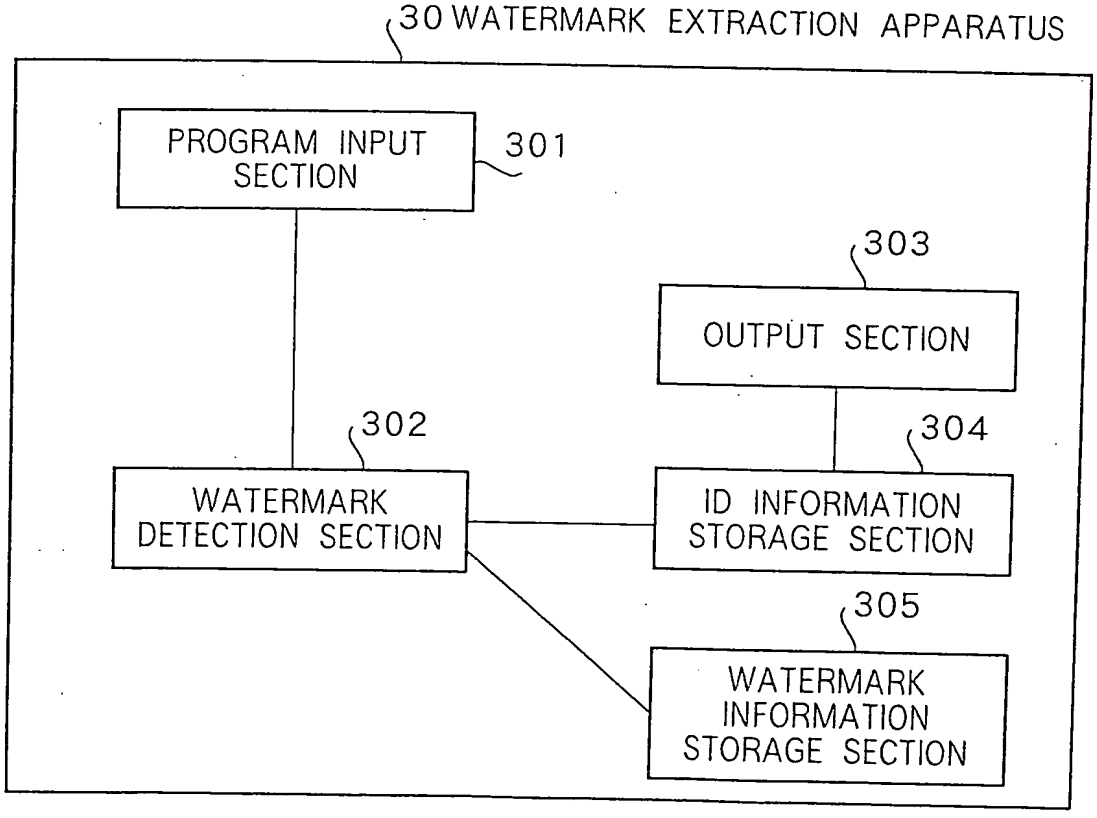
30 WATERMARK EXTRACTION APPARATUS
PROGRAM INPUT SECTION
301
303
OUTPUT SECTION
302
304
WATERMARK DETECTION SECTION
ID INFORMATION STORAGE SECTION
305
WATERMARK INFORMATION STORAGE SECTION

FIG. 3

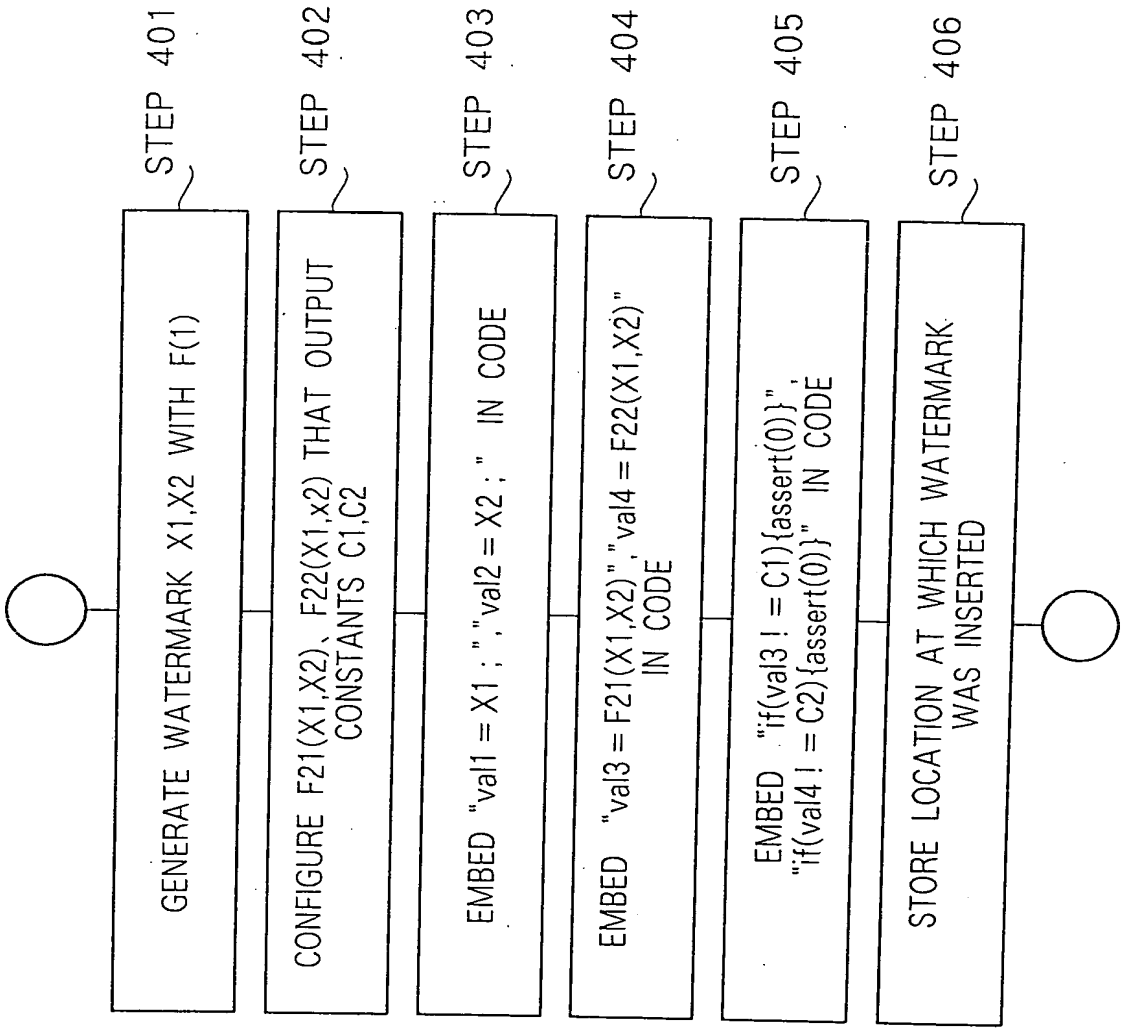
STEP 401
GENERATE WATERMARK X1,X2 WITH F(1)
STEP 402
CONFIGURE F21(X1,X2), F22(X1,x2) THAT OUTPUT CONSTANTS C1,C2
STEP 403
EMBED "val1 = X1 ; ","val2 = X2 ; " IN CODE
STEP 404
EMBED "val3 = F21(X1,X2)","val4 = F22(X1,X2)" IN CODE
STEP 405
EMBED "if(val3 ! = C1){assert(0)}", "if(val4 ! = C2){assert(0)}" IN CODE
STEP 406
STORE LOCATION AT WHICH WATERMARK WAS INSERTED

FIG. 4

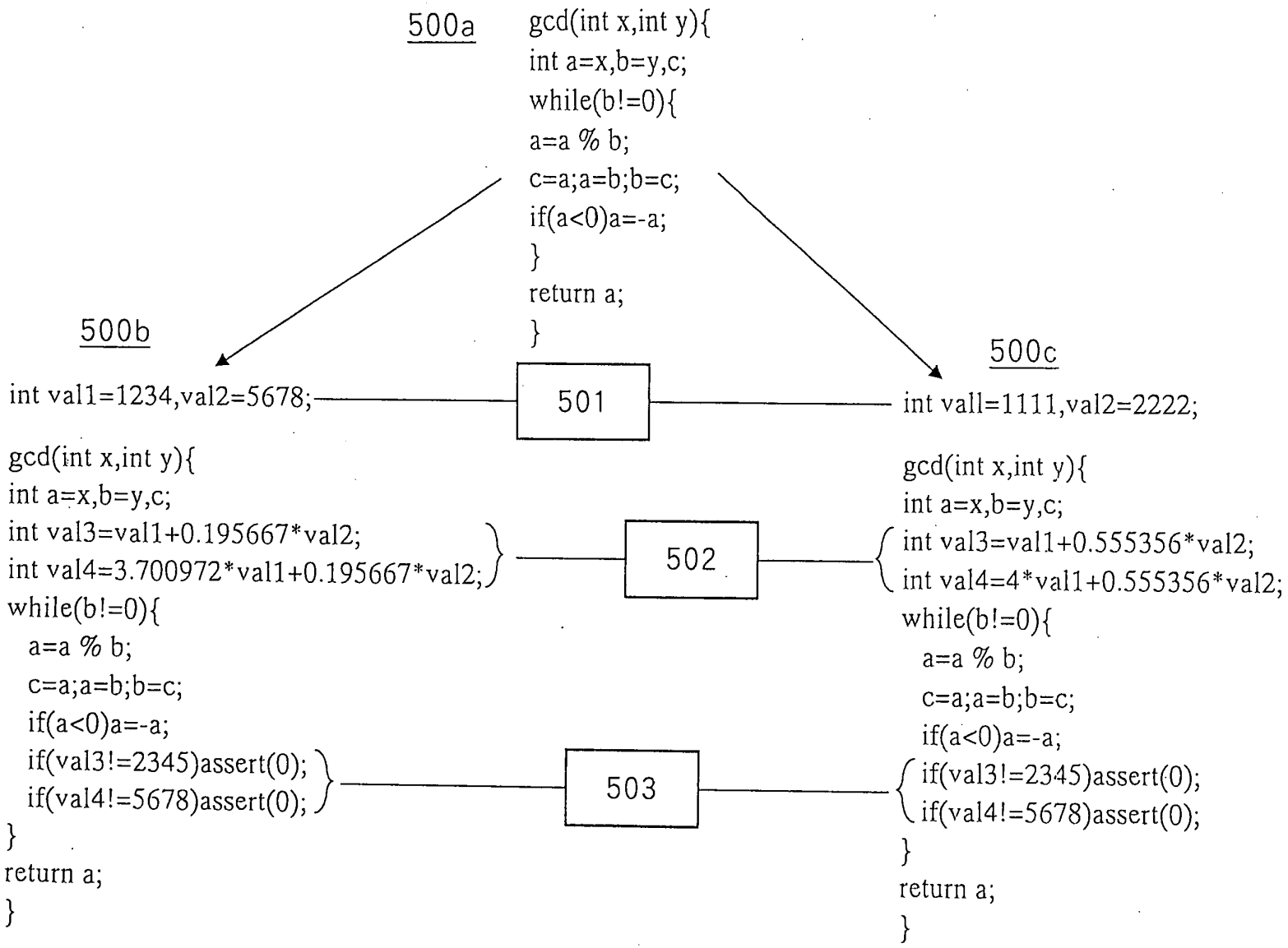
500a
gcd(int x,int y){
int a=x,b=y,c;
while(b!=0){
a=a % b;
c=a;a=b;b=c;
if(a<0)a=-a;
}
return a;
}
500b
int val1=1234,val2=5678;
gcd(int x,int y){
int a=x,b=y,c;
int val3=val1+0.195667*val2;
int val4=3.700972*val1+0.195667*val2;
while(b!=0){
a=a % b;
c=a;a=b;b=c;
if(a<0)a=-a;
if(val3!=2345)assert(0);
if(val4!=5678)assert(0);
}
return a;
}
501
502
503
500c
int val1=1111,val2=2222;
gcd(int x,int y){
int a=x,b=y,c;
int val3=val1+0.555356*val2;
int val4=4*val1+0.555356*val2;
while(b!=0){
a=a % b;
c=a;a=b;b=c;
if(a<0)a=-a;
if(val3!=2345)assert(0);
if(val4!=5678)assert(0);
}
return a;
}

FIG. 5

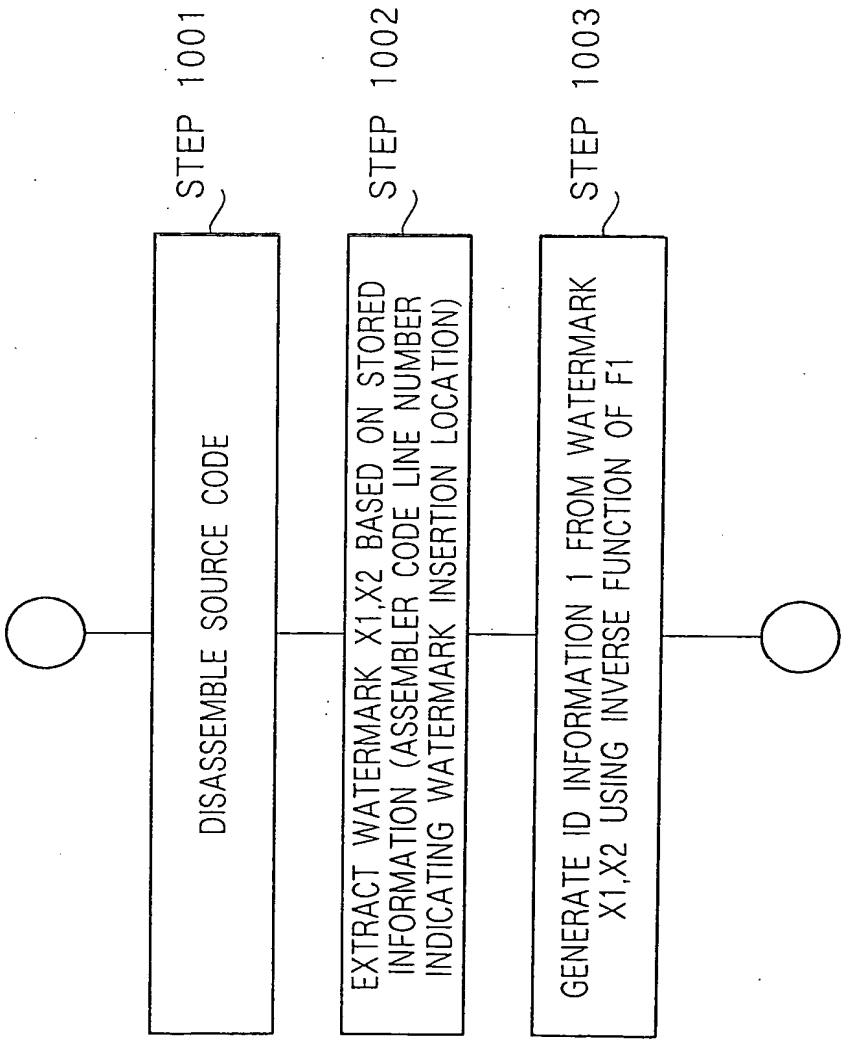
STEP 1001
DISASSEMBLE SOURCE CODE
STEP 1002
EXTRACT WATERMARK X1,X2 BASED ON STORED INFORMATION (ASSEMBLER CODE LINE NUMBER INDICATING WATERMARK INSERTION LOCATION)
STEP 1003
GENERATE ID INFORMATION 1 FROM WATERMARK X1,X2 USING INVERSE FUNCTION OF F1

FIG. 6

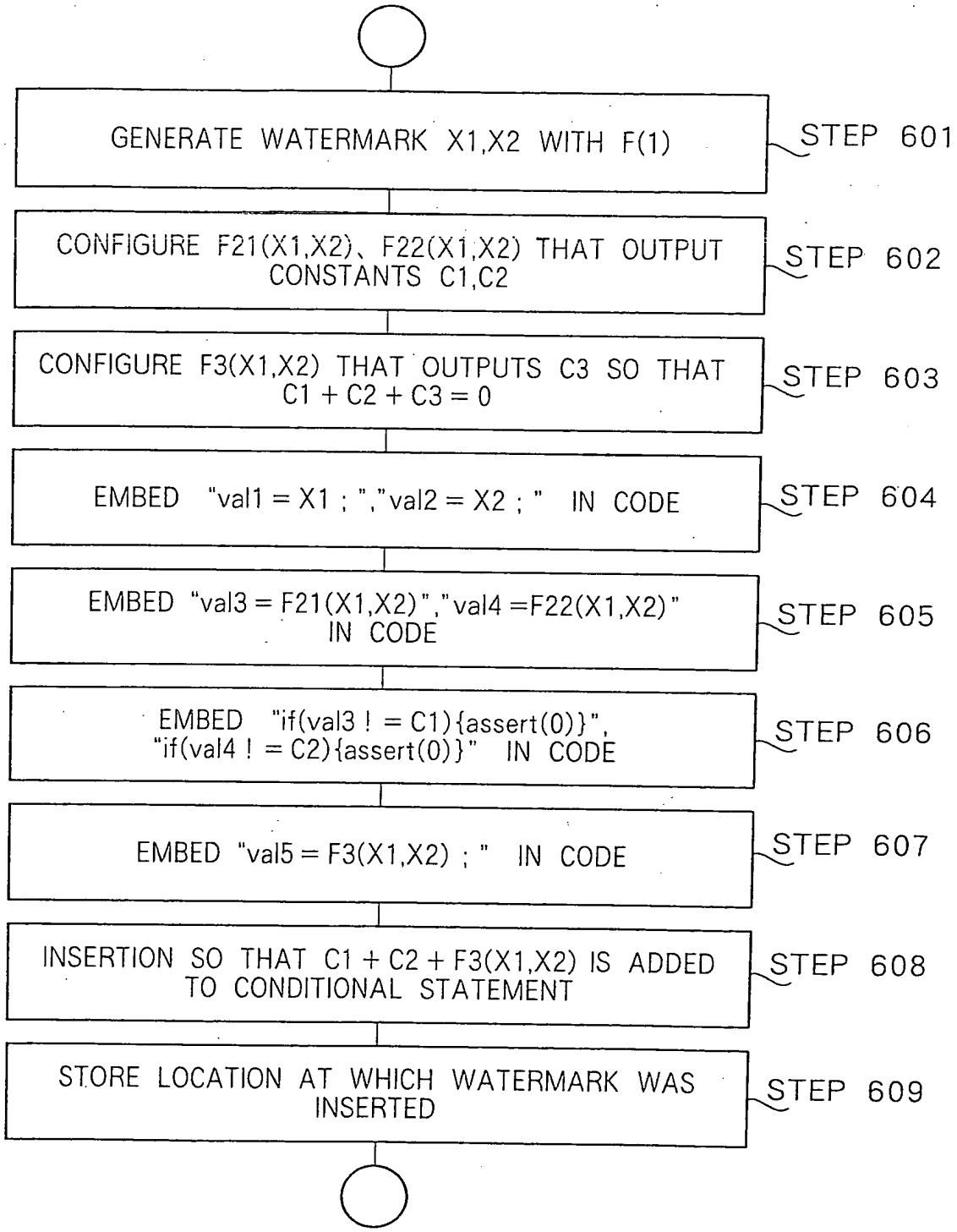
GENERATE WATERMARK X1,X2 WITH F(1)
STEP 601
CONFIGURE F21(X1,X2), F22(X1,X2) THAT OUTPUT CONSTANTS C1,C2
STEP 602
CONFIGURE F3(X1,X2) THAT OUTPUTS C3 SO THAT C1 + C2 + C3 = 0
STEP 603
EMBED "val1 = X1 ; ","val2 = X2 ; " IN CODE
STEP 604
EMBED "val3 = F21(X1,X2)","val4 =F22(X1,X2)" IN CODE
STEP 605
EMBED "if(val3 ! = C1){assert(0)}", "if(val4 ! = C2){assert(0)}" IN CODE
STEP 606
EMBED "val5 = F3(X1,X2) ; " IN CODE
STEP 607
INSERTION SO THAT C1 + C2 + F3(X1,X2) IS ADDED TO CONDITIONAL STATEMENT
STEP 608
STORE LOCATION AT WHICH WATERMARK WAS INSERTED
STEP 609

FIG. 7

800a

```
gcd(int x,int y){
int a=x,b=y,c;
while(b!=0){
a=a % b;
c=a; a=b; b=c;
if( a<0)a=-a;
}
return a;
}
```

↓

800b

```
int val1=1234,val2=5678;                          ~701

gcd(int x,int y){
int a=x, b=y, c;
int val3=val1+0.195667*val2;                      }
int val4=3.700972*val1+0.195667*val2;             } ~702
int val5=5.601297*val1+0.195667*val2;             }
while(b!=0){
  a=a % b;
  c=a;a=b;b=c;
  if(a<val3+val4+val5)a=-a;                       ~703
  if(val3!=2345)a++;                              }
  if(val4!=5678)assert(0);                        } ~704
}
return a;
}
```

FIG. 8

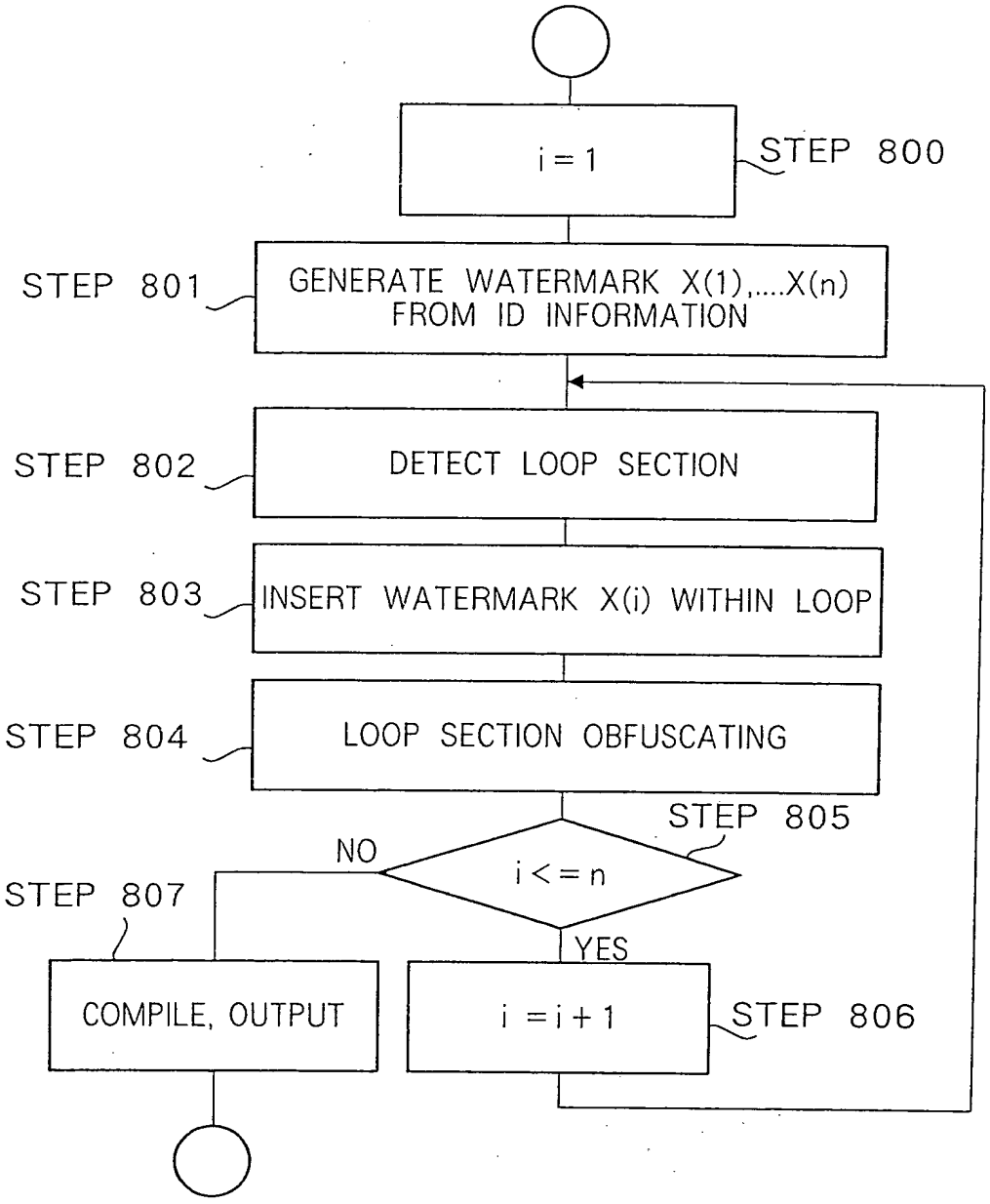
i = 1
STEP 800
STEP 801
GENERATE WATERMARK X(1),....X(n) FROM ID INFORMATION
STEP 802
DETECT LOOP SECTION
STEP 803
INSERT WATERMARK X(i) WITHIN LOOP
STEP 804
LOOP SECTION OBFUSCATING
STEP 805
NO
i <= n
STEP 807
YES
COMPILE, OUTPUT
i = i + 1
STEP 806

FIG. 9

900a

```
sample(int *A, int N){
  int i;
  int S=A[0];
  for(i =1;i<N;i++){
    if(S<A[i])S=A[i]
  {
   returs S;
}
```

900b

```
int val1;
sample(int *A, int N){
  int S =A[0];
  int i;
  while(1){
    val1=12345678;
    if(i>=N)break;
    if(S<A[i])S=A[i];
    i++;
    if(i>=N)break;
    if(S<A[i])S=A[i];
    i++;
  }
  return S;
}
```

902a

901

902b

900c

```
int val1;
sample(int *A, int N){
  int i=1;
  int S =A[0];
  while(i<N){
    if(i>=N)break;
    if(S<A[i])S=A[i];
    val1=11112222;
    N--;
    if(i<N)break;
    if(S<A[i])S=A[N];
    i++;
  }
  return S;
}
```

FIG.10

10/521789

10/521789

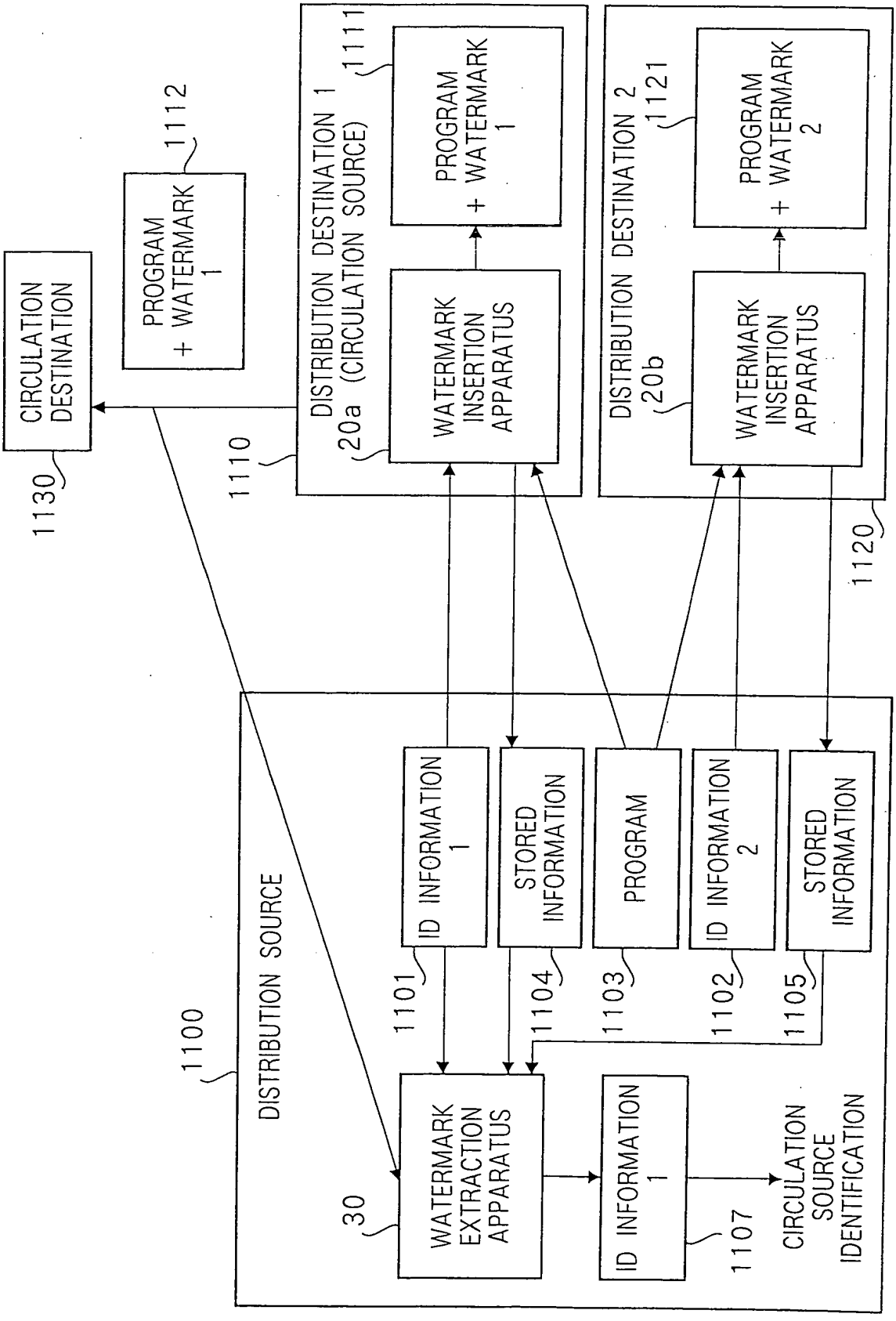

FIG.11

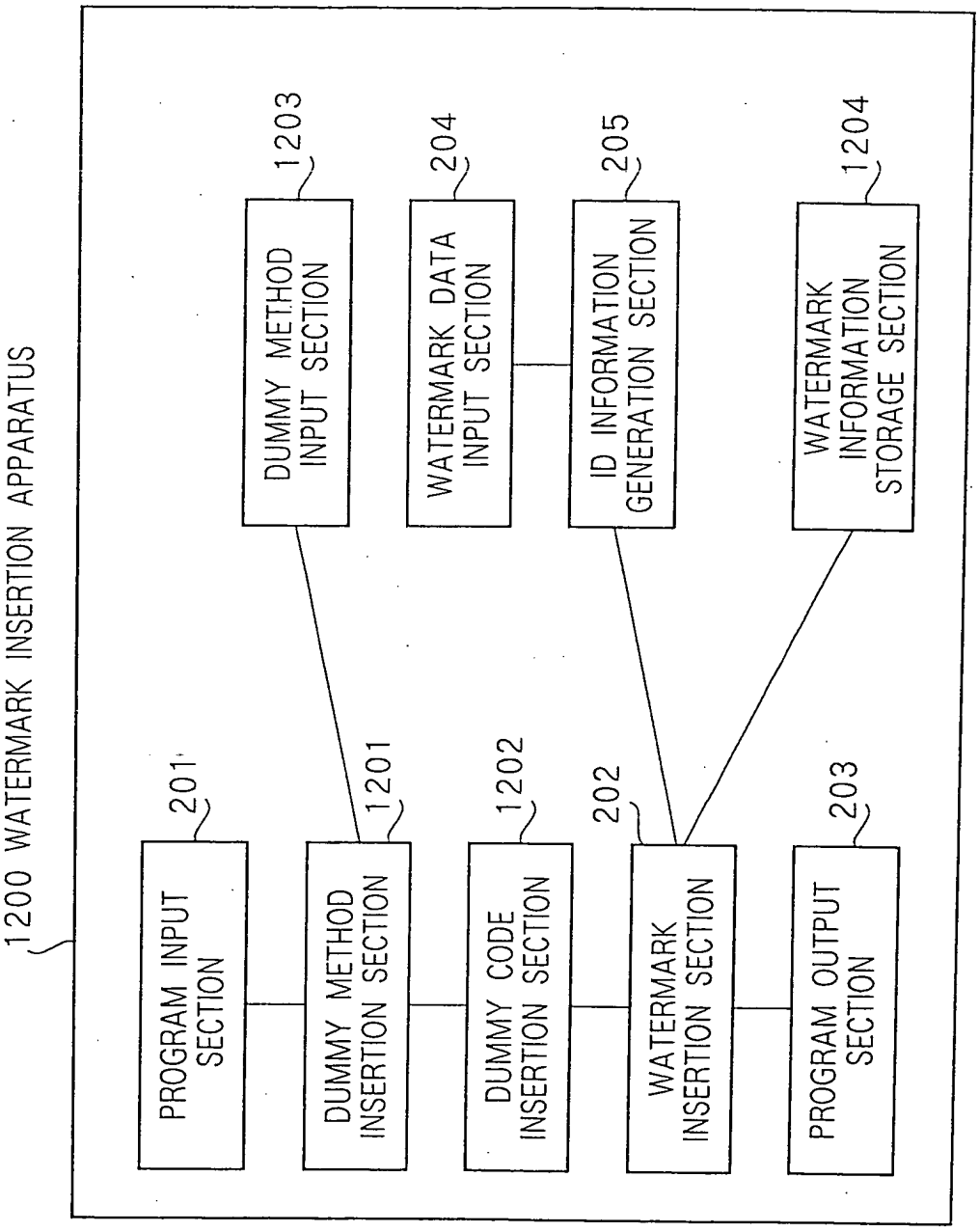
1200 WATERMARK INSERTION APPARATUS
PROGRAM INPUT SECTION
201
DUMMY METHOD INSERTION SECTION
1201
DUMMY CODE INSERTION SECTION
1202
WATERMARK INSERTION SECTION
202
PROGRAM OUTPUT SECTION
203
DUMMY METHOD INPUT SECTION
1203
WATERMARK DATA INPUT SECTION
204
ID INFORMATION GENERATION SECTION
205
WATERMARK INFORMATION STORAGE SECTION
1204

FIG.12

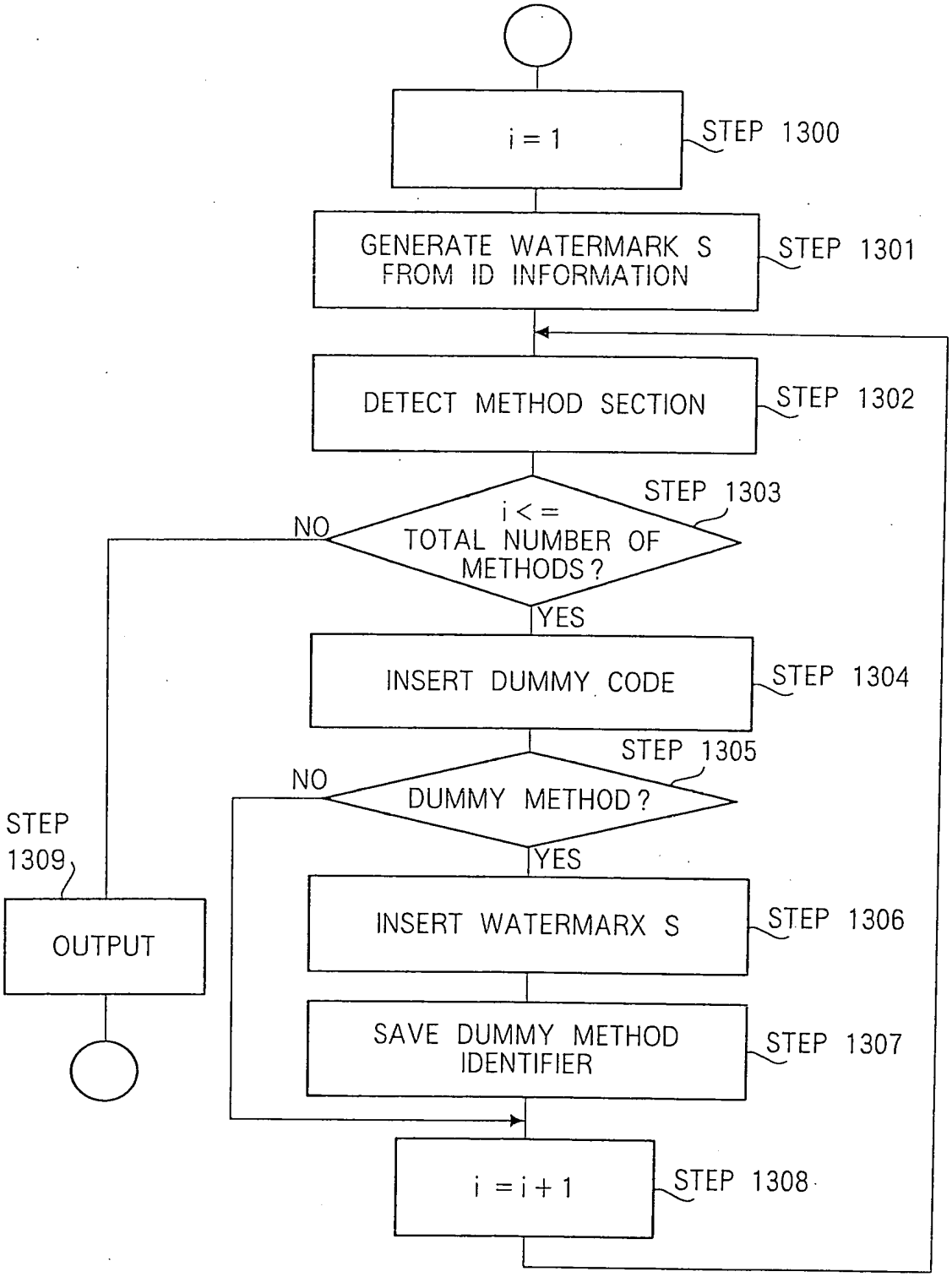
i = 1
STEP 1300
GENERATE WATERMARK S FROM ID INFORMATION
STEP 1301
DETECT METHOD SECTION
STEP 1302
i <= TOTAL NUMBER OF METHODS ?
STEP 1303
NO
YES
INSERT DUMMY CODE
STEP 1304
DUMMY METHOD ?
STEP 1305
NO
YES
INSERT WATERMARX S
STEP 1306
SAVE DUMMY METHOD IDENTIFIER
STEP 1307
i = i + 1
STEP 1308
STEP 1309
OUTPUT

FIG.13

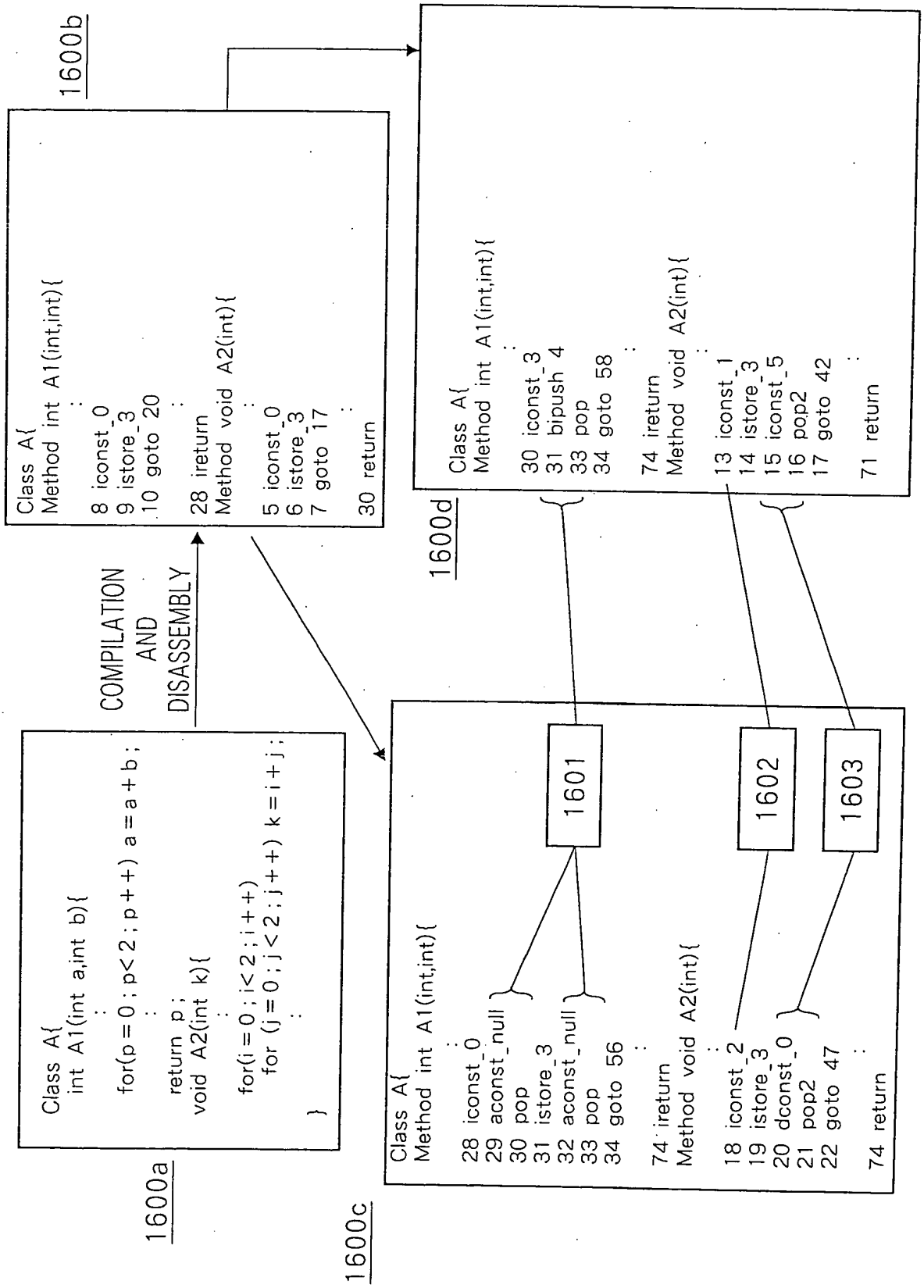
1600a
Class A{
int A1(int a,int b){
for(p = 0 ; p< 2 ; p++) a = a + b ;
return p ;
void A2(int k){
for(i = 0 ; i< 2 ; i++)
for (j = 0 ; j < 2 ; j++) k = i + j ;
}
COMPILATION AND DISASSEMBLY
1600b
Class A{
Method int A1(int,int){
8 iconst_0
9 istore_3
10 goto 20
28 ireturn
Method void A2(int){
5 iconst_0
6 istore_3
7 goto 17
30 return
1600d
Class A{
Method int A1(int,int){
30 iconst_3
31 bipush 4
33 pop
34 goto 58
74 ireturn
Method void A2(int){
13 iconst_1
14 istore_3
15 iconst_5
16 pop2
17 goto 42
71 return
1600c
Class A{
Method int A1(int,int){
28 iconst_0
29 aconst_null
30 pop
31 istore_3
32 aconst_null
33 pop
34 goto 56
74 ireturn
Method void A2(int){
18 iconst_2
19 istore_3
20 dconst_0
21 pop2
22 goto 47
74 return
1601
1602
1603

FIG.14

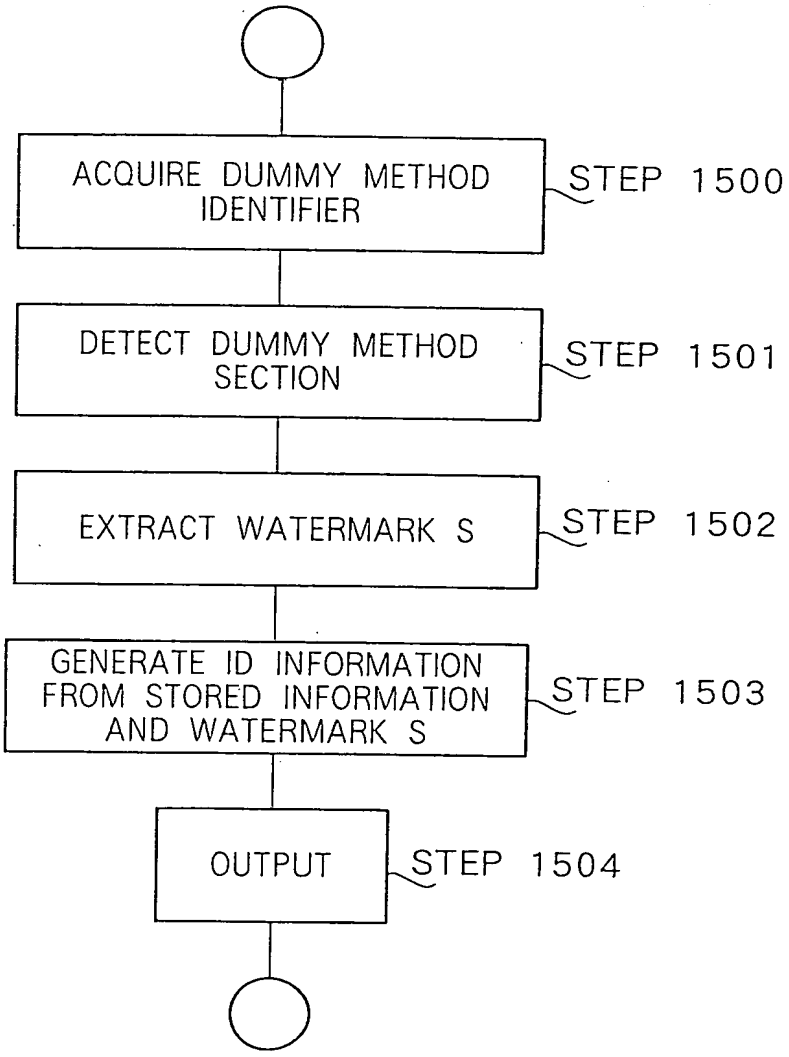
ACQUIRE DUMMY METHOD IDENTIFIER
STEP 1500
DETECT DUMMY METHOD SECTION
STEP 1501
EXTRACT WATERMARK S
STEP 1502
GENERATE ID INFORMATION FROM STORED INFORMATION AND WATERMARK S
STEP 1503
OUTPUT
STEP 1504

FIG.15

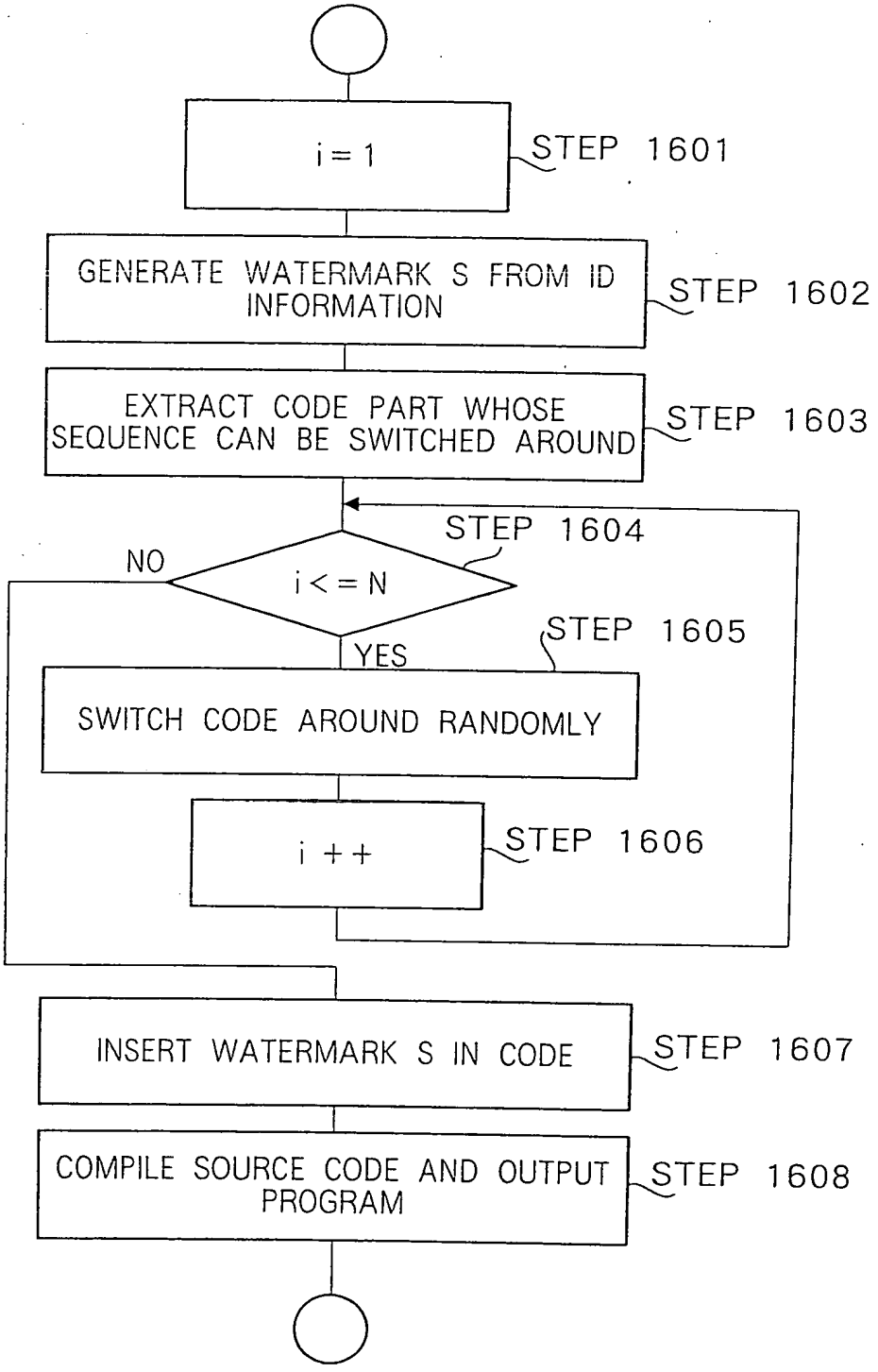
i = 1
STEP 1601
GENERATE WATERMARK S FROM ID INFORMATION
STEP 1602
EXTRACT CODE PART WHOSE SEQUENCE CAN BE SWITCHED AROUND
STEP 1603
STEP 1604
NO
i <= N
YES
STEP 1605
SWITCH CODE AROUND RANDOMLY
i ++
STEP 1606
INSERT WATERMARK S IN CODE
STEP 1607
COMPILE SOURCE CODE AND OUTPUT PROGRAM
STEP 1608

FIG.16

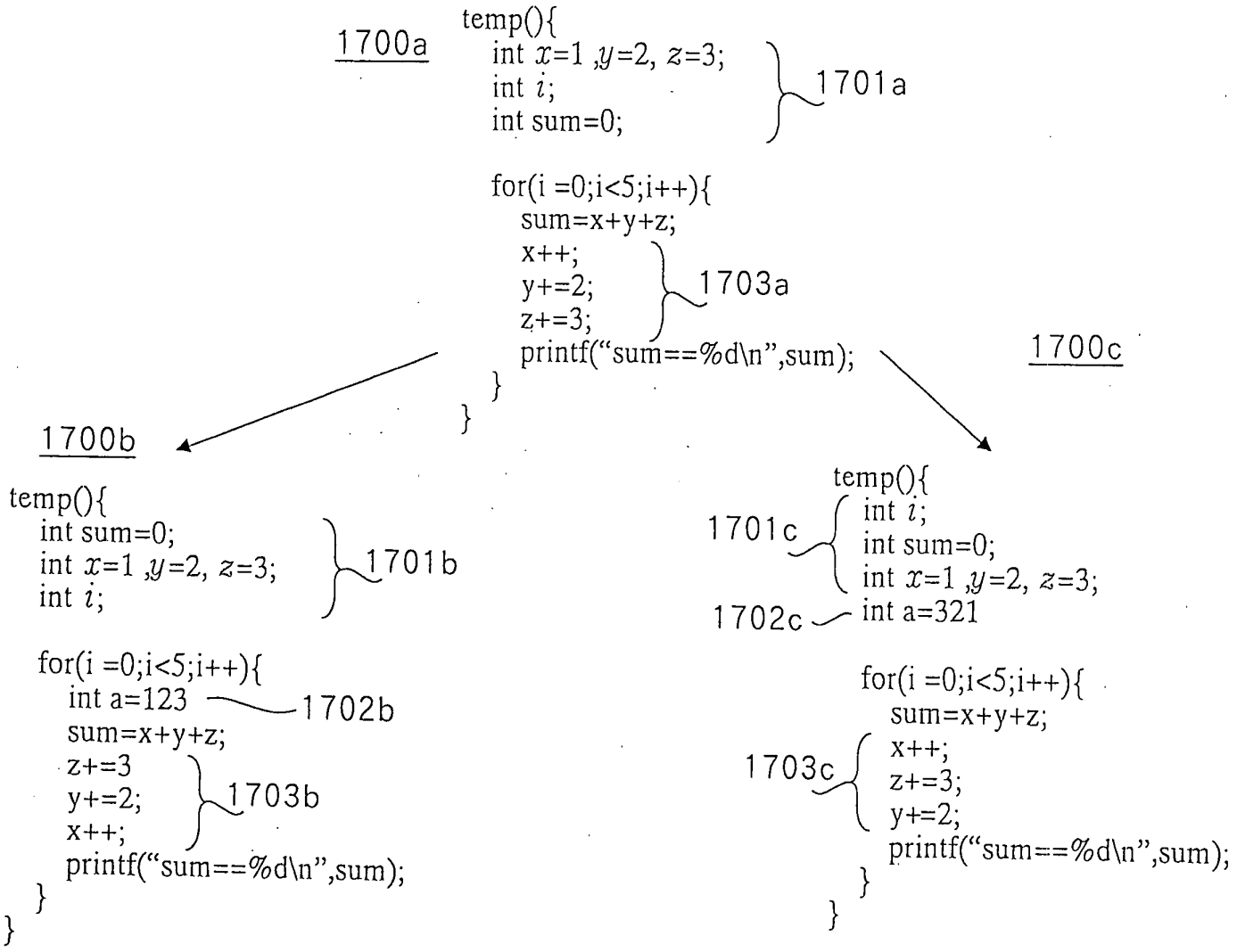
1700a
temp(){
int x=1 ,y=2, z=3;
int i;
int sum=0;
1701a
for(i =0;i<5;i++){
sum=x+y+z;
x++;
y+=2;
z+=3;
1703a
printf("sum==%d\n",sum);
}
}
1700b
temp(){
int sum=0;
int x=1 ,y=2, z=3;
int i;
1701b
for(i =0;i<5;i++){
int a=123
1702b
sum=x+y+z;
z+=3
y+=2;
x++;
1703b
printf("sum==%d\n",sum);
}
}
1700c
temp(){
1701c
int i;
int sum=0;
int x=1 ,y=2, z=3;
1702c
int a=321
for(i =0;i<5;i++){
sum=x+y+z;
1703c
x++;
z+=3;
y+=2;
printf("sum==%d\n",sum);
}
}

FIG.17